On the effect of variance-stabilizing transformation to wavelet-based software reliability assessment (Mathematical Models of Decision Making under Uncertainty and Ambiguity and Related Topics)

AUTHOR(S):

肖, 霄

CITATION:

肖, 霄. On the effect of variance-stabilizing transformation to wavelet-based software reliability assessment (Mathematical Models of Decision Making under Uncertainty and Ambiguity and Related Topics). 数理解析研究所講究録 2016, 1990: 244-251: KJ00010218550.

ISSUE DATE:

2016-04

URL:

http://hdl.handle.net/2433/224583

数理解析研究所講究録
第1990巻 2016年 244-251

On the effect of variance-stabilizing transformation to wavelet-based software reliability assessment

首都大学東京・システムデザイン研究科　肖　霄
Xiao Xiao
Graduate School of System Design,
Tokyo Metropolitan University

1 Introduction

In software reliability engineering, it has become one of the main issues in this area to assess the software reliability quantitatively. Among over hundreds of software reliability models (SRMs) [11, 12, 14], non-homogeneous Poisson process (NHPP) based SRMs have gained much popularity in actual software testing phases. People are interested in estimating the *software intensity function* of NHPP-based SRMs from software fault count data that can be collected in the software testing phases. The software intensity function in discrete time denotes the number of software faults detected per unit time. From the estimated software intensity function, it is possible to predict the number of remaining software faults and the quantitative software reliability, which is defined as the probability that software system does not fail during a specified time period under a specified operational environment. Therefore, we are interested in developing a high-accuracy estimation method for the software intensity function.

We proposed wavelet shrinkage estimation (WSE) as non-parametric estimation method for NHPP-based SRMs ([15, 16, 17]). The WSE does not require solving any optimization problem, so that the implementation of estimation algorithms is rather easy than the other non-parametric methods. We compared our method with the conventional maximum likelihood estimation (MLE) and the least squares estimation (LSE) through goodness-of-fit test. It has been shown that WSE could provide higher goodness-of-fit performance than MLE and LSE in many cases, in spite of its non-parametric nature, through numerical experiments with real software fault count data.

The fundamental idea of WSE is to remove the noise included in the observed software fault count data to get a noise-free estimate of the software intensity function. It is performed through the following three-step procedure. First, the noise variance is stabilized by applying the variance-stabilizing transformation to the data. This produces a time-series data in which the noise can be treated as Gaussian white noise. Second, the noise is removed using threshold methods. Third, an inverse variance-stabilizing transformation is applied to the denoised time-series data, obtaining the estimate of the software intensity function. This paper focuses on the first and the third steps of WSE and aims at identifying and emphasizing the influence that the data transformation exerts on the accuracy of WSE. The remaining part of this paper is planned as follows. In Section 2, we give a preliminary on wavelet analysis, especially focusing on multiresolution representation. Section 3 describes the WSE for NHPP-based SRMs in details. In Section 4, we carry out the real project data analysis and examine the effectiveness of eight data transformations. Finally, the paper is concluded in Section 5.

2 Multiresolution Representation

Let $f(t)$ denote the target function on continuous time t. In multiresolution representation, target function is usually expressed by the linear combination of its *approximation component* and *detail component*, *i.e.*,

$$f(t) = f_{j_0}(t) + \sum_{j=j_0}^{+\infty} g_j(t), \tag{1}$$

where $f_{j_0}(t)$ and $g_j(t)$ are the level-j_0 approximation component and level-j detail component of $f(t)$, respectively. Here, parameter j (≥ 0) is called resolution level, and j_0 is primary resolution level. The level-j approximation component $f_j(t)$ and level-j detail $g_j(t)$ component are defined as follows.

$$f_j(t) = \sum_{k=-\infty}^{+\infty} \alpha_{j,k}\phi_{j,k}(t), \tag{2}$$

$$g_j(t) = \sum_{k=-\infty}^{+\infty} \beta_{j,k}\psi_{j,k}(t). \tag{3}$$

Here, $\alpha_{j,k}$ and $\beta_{j,k}$ are the so-called *scaling coefficients* and *wavelet coefficients*, respectively. $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ are wavelet bases that are generated from *father wavelet* and *mother wavelet*, respectively. If we choose Haar wavelet, of which the father wavelet and mother wavelet are given by

$$\phi(t) = \begin{cases} 1 & (0 \leq t \leq 1) \\ 0 & (\text{otherwise}), \end{cases} \tag{4}$$

and

$$\psi(t) = \begin{cases} 1 & (0 \leq t < 1/2) \\ -1 & (1/2 \leq t < 1), \\ 0 & (\text{otherwise}) \end{cases} \tag{5}$$

then $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ can be generated by introducing a scaling parameter j and a shift parameter k as

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k) = \begin{cases} 2^{j/2} & (2^{-j}k \leq t \leq 2^{-j}(k+1)) \\ 0 & (\text{otherwise}), \end{cases} \tag{6}$$

and

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) = \begin{cases} 2^{j/2} & (2^{-j}k \leq t < 2^{-j}(k+1/2)) \\ -2^{j/2} & (2^{-j}(k+1/2) \leq t < 2^{-j}(k+1)) \\ 0 & (\text{otherwise}). \end{cases}$$

Because $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ are orthonormal bases, their coefficients can be found by calculating the inner product of the target function and themselves. Therefore, the scaling coefficients and wavelet coefficients are defined as

$$\alpha_{j,k} = \int_{-\infty}^{\infty} f(t)\phi_{j,k}^*(t)dt, \tag{7}$$

$$\beta_{j,k} = \int_{-\infty}^{\infty} f(t)\psi_{j,k}^*(t)dt, \tag{8}$$

where notation "*" means complex conjugation.

In practice, the highest resolution level J depends on the length of observed data, say n ($= 2^J$). Additionally, the primary resolution level j_0 is often set to be 0. Therefore, the level-J multiresolution representation of $f(t)$ is

given by

$$f_J(t) = f_0(t) + \sum_{j=0}^{J-1} g_j(t), \tag{9}$$

$$f_j(t) = \sum_{k=0}^{2^j-1} \alpha_{j,k}\phi_{j,k}(t), \tag{10}$$

$$g_j(t) = \sum_{k=0}^{2^j-1} \beta_{j,k}\psi_{j,k}(t). \tag{11}$$

In other words, the level-J multiresolution representation of $f(t)$ on continuous time t is given by

$$f_J(t) = \alpha_{0,0}\phi_{0,0}(t) + \sum_{j=0}^{J-1}\sum_{k=0}^{2^j-1} \beta_{j,k}\psi_{j,k}(t). \tag{12}$$

On the other hand, in discrete time case, target function f_i ($i = 1, 2, \cdots, n$) can be expanded in a similar fasion:

$$f_i^J = u_{0,0}\phi_{0,0}(i) + \sum_{j=0}^{J-1}\sum_{k=0}^{2^j-1} v_{j,k}\psi_{j,k}(i), \tag{13}$$

where $u_{j,k}$ and $v_{j,k}$ are *discrete scaling coefficients* and *discrete wavelet coefficients*, respectively. Because the approximation error between continuous and discrete coefficients can be adjusted by factor $\sqrt{n}$ ([1]), we have

$$u_{j,k} = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} f_i\, \phi^*_{j,k}(i), \tag{14}$$

$$v_{j,k} = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} f_i\, \psi^*_{j,k}(i). \tag{15}$$

The mapping from function f_i to coefficients ($u_{j,k}$, $v_{j,k}$) is called the *discrete Haar wavelet transform* (DHWT), while the reconstruction of function f_i from coefficients ($u_{j,k}$, $v_{j,k}$) is called the *inverse discrete Haar wavelet transform* (IDHWT).

From inherence property of Haar wavelet, $u_{j,k}$ and $v_{j,k}$ can be calculated easily as follows.

For $j = J$ ($J = \log_2 n$), $k = 0, 1, \cdots, n-1$,

$$u_{j,k} = \frac{1}{\sqrt{n}} f_{k+1}. \tag{16}$$

For $j = J-1, J-2, \cdots, 0$, $k = 0, 1, \cdots, 2^j - 1$,

$$u_{j,k} = \frac{1}{\sqrt{2}}(u_{j+1,2k} + u_{j+1,2k+1}), \tag{17}$$

$$v_{j,k} = \frac{1}{\sqrt{2}}(u_{j+1,2k} - u_{j+1,2k+1}). \tag{18}$$

Furthermore, for $j = 0, 1, \cdots, J-1$, $k = 0, 1, \cdots, 2^j - 1$, the IDHWT is achieved by

$$u_{j+1,2k} = \frac{1}{\sqrt{2}}(u_{j,2k} + v_{j,2k+1}), \tag{19}$$

$$u_{j+1,2k+1} = \frac{1}{\sqrt{2}}(u_{j,2k} - v_{j,2k+1}). \tag{20}$$

Let s_i be the observation of f_i ($i = 1, 2, \ldots, n$). Then, *the empirical discrete scaling coefficients* $c_{j,k}$ and *empirical discrete wavelet coefficients* $d_{j,k}$ can be calculated using equations (14) and (15) with f_i replaced by s_i.

Coefficients $c_{j,k}$ and $d_{j,k}$ are called the empirical ones because it is considered that observation errors are included inside. That is, the noises of the data are considered to be involved in empirical discrete wavelet coefficients $d_{j,k}$, and the *threshold methods* can be used for denoising. Up to now, although a broad class of thresholding schemes are available in the literature, the common choices include *hard thresholding* and *soft thresholding*, where hard thresholding is a 'keep' or 'kill' rule, while soft thresholding is a 'shrink' or 'kill' rule. The hard thresholding is defined as

$$\delta_\tau(d) \quad = \quad d1_{|d|>\tau}, \tag{21}$$

and soft thresholding is defined as

$$\delta_\tau(d) \quad = \quad \mathrm{sgn}(d)(|d| - \tau)_+, \tag{22}$$

for a fixed threshold level τ (> 0), where 1_A is the indicator function of an event A, $\mathrm{sgn}(d)$ is the sign function of d, and $(d)_+ = \max(0, d)$. Letting $d''_{j,k}$ denote the denoised empirical discrete wavelet coefficients, the estimates of target function f_i can be obtained by applying IDHWT to $c_{j,k}$ and $d''_{j,k}$.

3 Wavelet Shrinkage Estimation for NHPP-based SRMs

Suppose that the number of software faults detected through a system test is observed at discrete time $i = 0, 1, 2, \ldots$. Let Y_i, and $N_i = \sum_{k=0}^{i} Y_k$ respectively denote the number of software faults detected at testing date i, and its cumulative value, where $Y_0 = N_0 = 0$ is assumed without any loss of generality. The stochastic process $\{N_i : i = 0, 1, 2, \ldots\}$ is said to be a discrete non-homogeneous Poisson process (D-NHPP) if the probability mass function at time i is given by

$$\Pr\{N_i = m\} \quad = \quad \frac{\{\Lambda_i\}^m}{m!}\exp\{-\Lambda_i\}, \quad m = 0, 1, 2, \ldots, \tag{23}$$

where $\Lambda_i = E[N_i]$ is the expected cumulative number of software faults detected by testing date i. The function $\lambda_i = \Lambda_i - \Lambda_{i-1}$ ($i \geq 1$) is called the discrete intensity function, and implies the expected number of faults detected at testing date i, say $\lambda_i = E[Y_i]$. In other words,

$$Y_i \quad = \quad \lambda_i + \eta_i, \quad i = 0, 1, 2 \ldots. \tag{24}$$

The wavelet-based techniques have been well established especially in several areas such as non-parametric regression, probability density estimation, time-series analysis, *etc.* Consider the following non-parametric regression model:

$$S_i \quad = \quad z_i + \epsilon_i, \quad i = 1, 2, \ldots, n, \tag{25}$$

where z_i are arbitrary target functions of discrete-time index i and ϵ_i are independent Gaussian random variables with $\mathrm{N}(0, \sigma^2)$ and σ (> 0). One of the basic approaches to the Gaussian non-parametric regression is to consider the unknown function z_i expanded as a wavelet series and to transform the original problem to an estimation of the wavelet coefficients from the data. Donoho and Johnstone [6, 7, 8] and Donoho *et al.* [9] proposed non-linear wavelet estimators of z_i, and suggested the extraction of the significant wavelet coefficients by *thresholding*, where wavelet coefficients are set to 0 if their absolute value is below a certain threshold level. Then, application of the inverse wavelet transform of denoised coefficients by thresholding leads to an estimator of the underlying target function z_i.

This standard wavelet-based technique can be applied to a non-parametric estimation of the discretized NHPP. The basic idea is to pre-process the Poisson count data using normalizing and variance-stabilizing transforms

Table 1: Representative Data Transformaions (the case of variance equals to 1/4).

	Data Transformation	Inverse Data Transformation	Distribution of Random Variable after DT
BT1 ([3, 4])	$B_i = \sqrt{Y_i}$	$Y_i = (B_i)^2$	$N(\sqrt{\lambda_i},\ 1/4)$
BT2 ([3, 4])	$B_i = \sqrt{Y_i + 1/2}$	$Y_i = (B_i)^2 - 1/2$	$N(\sqrt{\lambda_i + 1/2},\ 1/4)$
AT ([2])	$A_i = \sqrt{Y_i + 3/8}$	$Y_i = (A_i)^2 - 3/8$	$N(\sqrt{\lambda_i + 3/8},\ 1/4)$
FT ([10])	$F_i = 1/2 * (\sqrt{Y_i + 1} + \sqrt{Y_i})$	$Y_i = (F_i)^2 + 1/16 * (F_i)^{-2} - 1/2$	$N(1/2\{\sqrt{\lambda_i} + \sqrt{\lambda_i + 3/4}\},\ 1/4)$

Table 2: Representative Data Transformaions (the case of variance equals to 1).

	Data Transformation	Inverse Data Transformation	Distribution of Random Variable after DT
BT1 ([3, 4])	$B_i = 2\sqrt{Y_i}$	$Y_i = 1/4(B_i)^2$	$N(2\sqrt{\lambda_i - 1/4},\ 1)$
BT2 ([3, 4])	$B_i = 2\sqrt{Y_i + 1/2}$	$Y_i = 1/4((B_i)^2 - 2)$	$N(2\sqrt{\lambda_i + 1/4},\ 1)$
AT ([2])	$A_i = 2\sqrt{Y_i + 3/8}$	$Y_i = 1/4((A_i)^2 - 3/2)$	$N(2\sqrt{\lambda_i + 1/8},\ 1)$
FT ([10])	$F_i = \sqrt{Y_i + 1} + \sqrt{Y_i}$	$Y_i = 1/4((F_i)^2 + (F_i)^{-2} - 2)$	$N(2\sqrt{\lambda_i},\ 1)$

([3, 4, 2, 10]). Table 1 and Table 2 show representive variance-stabilizing transforms with different variances. By using either of them, the software fault count data, say y_i, which is the observation of Y_i ($i = 1, 2, \ldots, n$), is approximately transformed to the Gaussian data. That is, the transformed realizations s_i ($i = 1, 2, \ldots, n$) by data transformations can be regarded as the ones from the normally distributed random variables:

$$S_i = \lambda_i' + \nu_i, \quad i = 1, 2, \ldots, n, \tag{26}$$

where λ_i' is the transformed software intensity function, and ν_i is the Gaussian white noise with constant variance. Then, λ_i' is the target function to be expanded by multiresolution representation introduced in Section 2.

For thresholding, too large threshold may cut off important parts of the original function, whereas too small threshold retains noise in the selective reconstruction. Hence, the choice of threshold is also a significant problem. In fact many methods to determine the threshold level have been proposed (*e.g.* see [5]). In WSE ([15, 16, 17]), We use the universal threshold [6], and the 'leave-out-half' cross-validation threshold [13]:

$$\tau = \frac{\nu}{\sqrt{n}}\sqrt{2\log n}, \tag{27}$$

$$\tau = \left(1 - \frac{\log 2}{\log n}\right)^{-1/2}\tau\left(\frac{n}{2}\right), \tag{28}$$

where ν is the standard variation of empirical wavelet coefficients, and n is the length of the observation y_i. Refer to [13] for details of $\tau(n/2)$.

4 Real Data Analysis

Although eight kinds of variance-stabilizing transformations were used in WSE, the most appropriate one for NHPP-based SRMs was not be identified. In this experiment, we look into this problem. We use six sets of software fault count data (group data) collected from real project data sets ([11]). Let (K, n, x_n) denote the triple of the software size, the final testing date and the total number of detected fault. Then these data sets (DS1 ~ DS6)

Table 3: Goodness-of-fit test (DS1).

(i) $\nu^2 = 1/4$.

BT1		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.529866	0.149661	-79.726549
(h, cvt)	$J = 3$	3.025397	0.197381	-94.789276
(s, ut)	$J = 3$	3.446892	0.248472	-106.910033
(h, cvt)	$J = 3$	2.959304	0.188633	-92.269707
BT2		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.625475	0.188039	-94.180572
(h, cvt)	$J = 3$	1.686833	0.186644	-93.140641
(s, ut)	$J = 3$	2.134523	0.257615	-113.611754
(h, cvt)	$J = 3$	1.621912	0.172003	-90.456650
AT		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.604442	0.184318	-91.878435
(h, cvt)	$J = 3$	1.829843	0.187641	-93.666231
(s, ut)	$J = 3$	2.274154	0.256187	-112.792371
(h, cvt)	$J = 3$	1.745720	0.173040	-90.386588
FT		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.834515	0.177085	-89.358961
(h, cvt)	$J = 3$	2.407003	0.199900	-95.730954
(s, ut)	$J = 3$	2.822620	0.255477	-111.645519
(h, cvt)	$J = 3$	2.297144	0.180016	-90.741903

(ii) $\nu^2 = 1$.

BT1		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.529866	0.149661	-79.726549
(h, cvt)	$J = 3$	3.098158	0.203413	-96.082059
(s, ut)	$J = 3$	3.446892	0.248472	-106.910033
(h, cvt)	$J = 3$	2.959304	0.188633	-92.269707
BT2		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.625475	0.188039	-94.180572
(h, cvt)	$J = 3$	1.694119	0.186190	-93.691947
(s, ut)	$J = 3$	2.134523	0.257615	-113.611754
(h, cvt)	$J = 3$	1.628012	0.173098	-90.618897
AT		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.604442	0.184318	-91.878435
(h, cvt)	$J = 3$	1.873377	0.192463	-94.645453
(s, ut)	$J = 3$	2.274154	0.256187	-112.792371
(h, cvt)	$J = 3$	1.751474	0.174046	-90.534426
FT		MSE1	MSE2	LL
(h, ut)	$J = 3$	1.834515	0.177085	-89.358961
(h, cvt)	$J = 3$	2.378928	0.197520	-94.988778
(s, ut)	$J = 3$	2.822620	0.255477	-111.645519
(h, cvt)	$J = 3$	2.297144	0.180016	-90.741903

are presented by (14*kloc*, 62*weeks*, 133), (15*kloc*, 41*weeks*, 351), (103*kloc*, 73*weeks*, 367), (*null*, 46*days*, 266), (300*kb*, 81*days*, 461), (200*kloc*, 111*days*, 481), respectively. The MSE (mean squares error) and LL (log likelihood) are employed as the goodness-of-fit measures, where

$$\mathrm{MSE}_1 = \frac{\sqrt{\sum_{i=1}^{n}(\Lambda_i - x_i)^2}}{n}, \tag{29}$$

$$\mathrm{MSE}_2 = \frac{\sqrt{\sum_{i=1}^{n}(\lambda_i - y_i)^2}}{n}, \tag{30}$$

$$\mathrm{LL} = \sum_{i=1}^{n}(x_i - x_{i-1})\ln[\Lambda_i - \Lambda_{i-1}] - \Lambda_n - \sum_{i=1}^{n}\ln[(x_i - x_{i-1})!]. \tag{31}$$

We apply the WSE with two thresholding techniques (hard thresholding (h) v.s. soft thresholding (s)), and two thresholds (universal threshold (ut) v.s. cross-validation threshold (cvt)). Table 3 presents the goodness-of-fit results of DS1, based on the four threshold methods for DS1 with different variance-stabilizing transformations. Due to page limit, the results shown here are of the case where the highest resolution level J is set to be 3.

For all the variance-stabilizing transformations, it is observed that when universal threshold is used, the accuracies are the same between hard thresholding and soft thresholding. While in the case of cross-validation threshold,

the accuracy of the case of $\nu^2 = 1/4$ is the same with or better than that of the case of $\nu^2 = 1$ in most cases. The reasons are considered to be as follows. For the case of universal threshold, because the magnitude relationships between transformed data and the universal threshold are the same, the same empirical discrete wavelet coefficients $d_{j,k}$ are set to be 0. Therefore, the accuracies are the same from both theoretical and experimental points of view. That is, the variance of the variance-stabilizing transforms does not affect the accuracy of WSE when universal threshold is preferred. On the other hand, the same magnitude relationship does not exist in the case of cross-validation threshold. Hence, the accuracies are different when different variance-stabilizing transformations are used from theoretical point of view. However, it is observed in all the six data sets of our experiments that, the accuracy of the case of $\nu^2 = 1/4$ is the same with or better than that of the case of $\nu^2 = 1$ in most cases. Therefore, a variance-stabilizing transformation with smaller variance is preferred.

5 Conclusion

In this paper, the effectiveness of variance-stabilizing transformations to wavelet shrinkage estimation was investigated. From the numerical evaluation, we found that the accuracy of the case of $\nu^2 = 1/4$ was the same with or better than that of the case of $\nu^2 = 1$ in most cases. Therefore, we concluded that such variance-stabilizing transformations that transform original variance to 1/4 should be used in wavelet shrinkage estimation to assess software reliability.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 26730039.

References

[1] F. Abramovich, T. C. Bailey, and T. Sapatinas, "Wavelet analysis and its statistical applications," *The Statistician*, **49** (1), pp. 1–29 (2000).

[2] F. J. Anscombe, "The transformation of Poisson, binomial and negative binomial data," *Biometrika*, vol. 35, no. 3-4, pp. 246–254, 1948.

[3] M. S. Bartlett, "The square root transformation in the analysis of variance," *Supplement to the Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936.

[4] M. S. Bartlett, "The use of transformation," *Biometrics*, vol. 3, no. 1, pp. 39–52, 1947.

[5] P. Besbeas, I. De Feis and T. Sapatinas, "A comparative simulation study of wavelet shrinkage estimators for Poisson counts," *International Statistical Review*, vol. 72, no. 2, pp. 209–237, 2004.

[6] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[7] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, vol. 90, pp. 1200–1224, 1995.

[8] D. L. Donoho and I. M. Johnstone, "Minimax estimation via wavelet shrinkage," *Annals of Statistics*, vol. 26, no. 3, pp. 879–921, 1998.

[9] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian and D. Ricard, "Wavelet shrinkage: asymptopia? (with discussion)," *Journal of the Royal Statistical Society: Series B*, vol. 57, pp. 301–337, 1995.

[10] M. F. Freeman and J. W. Tukey, "Transformations related to the angular and the square root," *The Annals of Mathematical Statistics*, vol. 21, no. 4, pp. 607–611, 1950.

[11] M. R. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996.

[12] J. D. Musa, A. Iannino and K. Okumoto, *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[13] G. P. Nason, "Wavelet shrinkage using cross-validation," *Journal of the Royal Statistical Society: Series B*, vol. 58, pp. 463–479, 1996.

[14] H. Pham, *Software Reliability*, Springer, Singapore, 2000.

[15] X. Xiao and T. Dohi, "Wavelet-based approach for estimating software reliability," *Proceedings of 20th International Symposium on Software Reliability Engineering* (ISSRE'09), pp. 11–20, IEEE CS Press, 2009.

[16] X. Xiao and T. Dohi, "A comparative study of data transformations for wavelet shrinkage estimation with application to software reliability assessment," *Advances in Software Engineering*, vol. 2012, Article ID 524636, 9 pages, May 2012.

[17] X. Xiao and T. Dohi, "Wavelet shrinkage estimation for non-homogenous Poisson process based software reliability models," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 211–225, 2013.